Handout 1: Practice RA/SQL Queries

CS 4604 (Prakash, Spring 2016)

January 25, 2016

Consider the following scenario modelling courses, students, professors, departments, and the like at a single university in a single semester.

1. Each student has a name, a unique PID, and an address. A professor has a name, a unique PID, and belongs to a department. We also want to record the age and office of the professor. Each course has a name, a number, an offering department, a classroom, and an enrollment. (This university has not yet invented the concept of CRNs.) Each department offers only one course with each number.

2. Each department has a unique name. Each department has at most one chairperson who is its head (there are times when a department may not have a chairperson). Each chairperson can be the head of at most one department.

3. Each student enrolls in a certain number of courses in the semester. At most one professor teaches each course. Each student receives a grade in each course he/she is enrolled in. In turn, each student evaluates the professor teaching the course.

4. A course can have multiple pre-requisites. A course can be a pre-requisites for multiple courses. A course cannot be a pre-requisite for itself! A student enrolled in a course must have enrolled in all its pre-requisites.

In class, we came up with the following relations (or a very similar set of relations) to model this scenario:

- `Students(PID: string, Name: string, Address: string)`
- `Professors(PID: string, Name: string, Office: string, Age: integer, DepartmentName: string)`
- `Courses(Number: integer, DeptName: string, CourseName: string, Classroom: string, Enrollment: integer)`
- `Departments(Name: string, ChairPID: string)`
- `Take(StudentPID: string, Number: integer, DeptName: string, Grade: string, ProfessorEvaluation: integer)`
- `Teach(ProfessorPID: string, Number: integer, DeptName: string)`
- `PreReq(Number: integer, DeptName: string, PreReqNumber: integer, PreReqDeptName: string)`

Write down solutions to the following questions *both* in relational algebra and in SQL:

1. What are the PIDs of the students whose name is "Suri"?

2. Which pairs of students live at the same address? It is enough to return the names of such student pairs.

3. Which departments have courses that have pre-requisites in other departments?

4. Compute the set of all courses that are their own pre-requisites. The purpose of this query is to ensure that the constraint "A course cannot be a pre-requisite for itself" holds in the database. Your query needs to return only the course number and department name.

5. What are the names and addresses of the students who are taking "CS4604"?

6. What are the courses (specified by course number and department name) that the head of the CS department is teaching?

7. Return the PID and names of any department head who teaches a course in another department?

8. Are there any students who are taking at least two courses taught by department heads? Identify these students by their PID and name.

9. Does the PreReq relation have cycles?

10. A relation `R` has one numeric attribute `A`. What is the largest number in `R`?

11. Which professors (specify PID, Name, and Department) earn salaries more than any department head?

12. Which professor (specify PID, Name, and Department) earns the highest salary in each department?

13. A relation `R` has one numeric attribute `A`. The *rank* of a tuple t in `R` is the number of tuples in `R` whose value in `A` is less than the value of `t` in A. This question deals with computing the ranks of the tuples in `R`.

    (a) What is the median tuple in `R`, i.e., if `R` contains n tuple, what is the tuple with rank $n/2$.

    (b) Compute the rank of each tuple in `R`.

14. A *Pythagorean triple* is a list of three positive integers (a, b, c) such that $a^2 + b^2 = c^2$. The name comes from Pythagoras theorem, where a, b, and c can be the sides of a right-angled triangle. Find all the Pythagorean triples such that $c \leq 10$.

15. Find the name of the professor who teaches "CS4604."

    (a) Write the query in relational algebra using a natural join.

    (b) Write the query in relational algebra using intersection. This version of the query has a counterpart in SQL that uses sub-queries.